BIOSECURITY RISK AND IMPACT CALCULATOR

by

SOMIL CHANDWANI

B.E., Rajiv Gandhi Technical University, Bhopal, 2006.

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2008

Approved by:

Major Professor
Dr. Daniel Andresen

Abstract

"BRIC" is a web survey application that can provide feedback to the feedyard managers regarding the different types of risk involved in their feedyards. By answering a set of basic questions in the survey, the application generates three categories of reports for the managers which provide them with measures to improve the existing condition of their feedyard. These dynamically generated reports can help to decrease the risk of introduction of some disease or its impact once it is introduced in a feedyard.

The survey can be beneficial to collect data from various feedyards through the internet. This collected data can be used to make some interesting analysis and beneficial conclusions in this field of research.

Table of Contents

List of Figures v
List of Tables vi
CHAPTER 1 - Introduction 1
1.1 Aim 1
1.2 Scope 1
1.3 Need of the Application 2
1.4 Related Work and Problems 2
1.5 Platform Specifications - Deployment 2
CHAPTER 2 - System Requirement Analysis 4
2.1 Information Gathering 4
2.2 System Feasibility 4
CHAPTER 3 - System Analysis 6
3.1 ER Diagram 6
3.2 Data Flow Diagrams 7
3.3 State Transition Diagram 8
CHAPTER 4 - Design 9
4.1 Design Goals 9
4.2Architectural Design 9
4.3 Procedural /Modular Approach 11
CHAPTER 5 - Implementation 12
5.1 Database Implementation 12
5.2 User Interface Design and Implementation 12
5.3 Technical Discussions 16
CHAPTER 6 - Testing 17
6.1 Unit Testing 17
6.2 Integration Testing 18
6.3 Validation Testing 18
6.4 White Box Testing 19

6.5 Stress and Performance Testing 19
CHAPTER 7 - Results 25
7.1 Biosecurity Practices 25
7.2 Biocontainment Practices 26
7.3 Security Practices 27
CHAPTER 8 - Conclusion 29
8.1 Limitations 29
8.2 Scope for Future Work 29
References 30

List of Figures

Figure 3.1 ER Diagram of the BRIC 6
Figure 3.2 Context Level Data Flow Diagram..... 7
Figure 3.3 Level 1 Data Flow Diagram 7
Figure 3.4 State Transition Diagram..... 8
Figure 4.1 Architectural Context Diagram 9
Figure 4.2 Component Diagram 10
Figure 5.1 Database Implementation using MS Access 12
Figure 5.2 Instructions provided at the beginning and on the click of help button. 13
Figure 5.3 Initial Information Screen, user ID is generated here..... 13
Figure 5.4 Validations used for compulsory questions..... 14
Figure 5.5 Menu has sub sections enabled only when the section has been completed. 14
Figure 5.6 Reports generated with a different Menu to view different reports and options. 15
Figure 5.7 Reports can be saved using Microsoft Excel or Adobe PDF options..... 15
Figure 6.1 While recording a script WAS monitors IE to capture all incoming data and stores it into a database file. 21
Figure 6.2 We can specify the number of simulated clients by setting the number of threads and number of sockets on each thread 21

List of Tables

Table 6-1 The following report was generated by the WAS tool for testing the performance..... 24
Table 7-1 Proportion of 106 feedyards in 5 Central Plains states at which practices relevant to biosecurity were implemented 26
Table 7-2 Proportion of 106 feedyards in 5 Central Plains states at which practices relevant to biocontainment were implemented. 27
Table 7-3 Proportion of 106 feedyards in 5 Central Plains states at which practices relevant to feedyard security were implemented. 28

Acknowledgements

I would like to express my deepest gratitude to Dr Mike Sanderson and Mr Joe Nisil, College of Veterinary Medicine, for their support, assistance and all other facilities that were required during the development of this application.

Heartiest thanks to Dr Daniel Andresen for his encouragement, guidance and his valuable advices during the course of my work.

I would like to thank Dr Gurdip Singh and Dr Mitchell Neilsen for serving in my committee and for their valuable cooperation during this project.

I would like to thank Aric Brandt, D.V.M., Kansas State University, for his patience and support for making me familiar with the terminologies used in the Veterinary Science.

CHAPTER 1 - Introduction

1.1 Aim

The aim of this application is to provide a platform which can provide feedback to the feedyard owners and managers. This feedback is to be provided in the form of preventive measures to improve the existing condition of their feedyards and minimize the existing risk. The application provides the following facilities:

1. A convenient and easy to use web survey interface which consists of basic set of questions. Answers to these questions can determine the risk involved in the feedyards.
2. A smart navigation system which allows the user to change the answers at any point of time if the section has been already completed.
3. Three dynamically generated and categorized reports based on the type of risk involved in the feedyards.
4. Exporting the reports to Excel or PDF to save it on the end user's machine.
5. Basic knowledge about the various diseases that might be a possibility within the feedyard animals or humans.
6. Basic principles which may decrease the risk of accidental disease introduction.

1.2 Scope

The application can be used for collecting surveys over a number of feedyards and help improve their existing condition. The data collected from the survey can prove very beneficial for the research involved in this field as it helps the researchers in this field to understand the current feedyard security practices. This data can be used to get an idea about the chances of a particular disease introduction in a feedyard and the cause of such a disease. Reports generated by the web application can be either used by the feedyard managers to minimize the risk in their feedyards or it can be used by researchers for comparison when making recommendations in these areas.

1.3 Need of the Application

There is little research data available in the area of feedyard biosecurity, biocontainment, and security. Objective data on real versus perceived risk is difficult to obtain for intentional disease introduction risks. Data about endemic agents in feedyards are more readily available but still limited. Some data is available on current practices in other animal production systems. More research is needed to understand which practices will be economically rewarding.

Gathering data through surveys is an effective way to determine what the current biosecurity practices are in feedyards. Surveys also help identify perceptions held by industry representatives about biological threats, routes of introduction, and the importance of mitigation strategies aimed at prevention or control. The results of this work will provide benchmarks for feedyard managers in the areas of biocontainment, biosecurity and security. Consulting veterinarians can use this information for comparison when making recommendations in these areas.

1.4 Related Work and Problems

There are various third party websites which do provide electronic surveys for users with a facility to add questions using their own template. The end users of our application are not very technical. Designing a survey application from the scratch makes it possible to develop a simple application. The responses to certain questions are dependent on other questions which can be answered only when a set of certain questions have already been answered. This dependency is decided dynamically and is not available in third party surveys. Categorizing the survey and the navigation from one part to another is not available in third party surveys. Also, the format of the report and its categorization is not available in third party surveys.

1.5 Platform Specifications - Deployment

1.5.1 Hardware Specification

Processor P III

RAM 128 MB

Minimum Space Required 10 MB

Display 16 bit color

1.5.2 Software Specification

Operating Environment Win 2000/XP

Platform .Net Framework & IIS

Database MS Access 2000

CHAPTER 2 - System Requirement Analysis

2.1 Information Gathering

For the development of this project I referred to the professors and students involved in research related to feedyard security practices including Dr Mike Sanderson, Associate Professor at College of Veterinary Medicine. As the survey should be convenient for the feedyard managers, I was provided with the details regarding important terminologies to be used in the application. The working of the survey is forced to be sequential and questions are categorized into specific sections to make it convenient and easy to use for the end user. Dr Sanderson is involved in the research of risk assessment for the feedyards and helped me to improve the user interface to a great extent as he is familiar with the view with which feedyard managers will look at the application. Dr Andresen, Associate Professor, CIS provided regular feedback and guided me to add more functionality to the project.

Other than this, I did a lot of research on various other survey applications which have already been implemented and was able to incorporate a few more stronger features into the application. The tools and controls used in the application are recommended ASP.NET controls by Microsoft which improves the navigation and report generation to a great extent.

2.2 System Feasibility

The system feasibility can be divided into the following sections:

2.2.1 Economic Feasibility

The project is economically feasible as the only cost involved is having a computer with the minimum requirements mentioned earlier. For the users to access the application and complete the survey, the only cost involved will be in getting access to the Internet.

2.2.2 Technical Feasibility

To deploy the application, the only technical aspects needed are mentioned below:

Operating Environment	Win 2000/XP
Platform	.Net Framework & IIS
Database	MS Access 2000

For Users:

Internet Browser

Internet Connection

2.2.3 Behavioral Feasibility

The application requires no special technical guidance and all the views available in the application are self explanatory. Also for any kind of queries, instructions are provided on the starting page of the application. A user can read these instructions at any point of time in the survey.

CHAPTER 3 - System Analysis

Working closely with the Dr Sanderson and analyzing the requirements and functionality of the web application, I had three important diagrams by the end of the analysis phase. The ER diagram, data flow diagram and the state transition diagram which were a basis for finding out entities and relationships between them, the flow of information and the various states the application can have.

3.1 ER Diagram

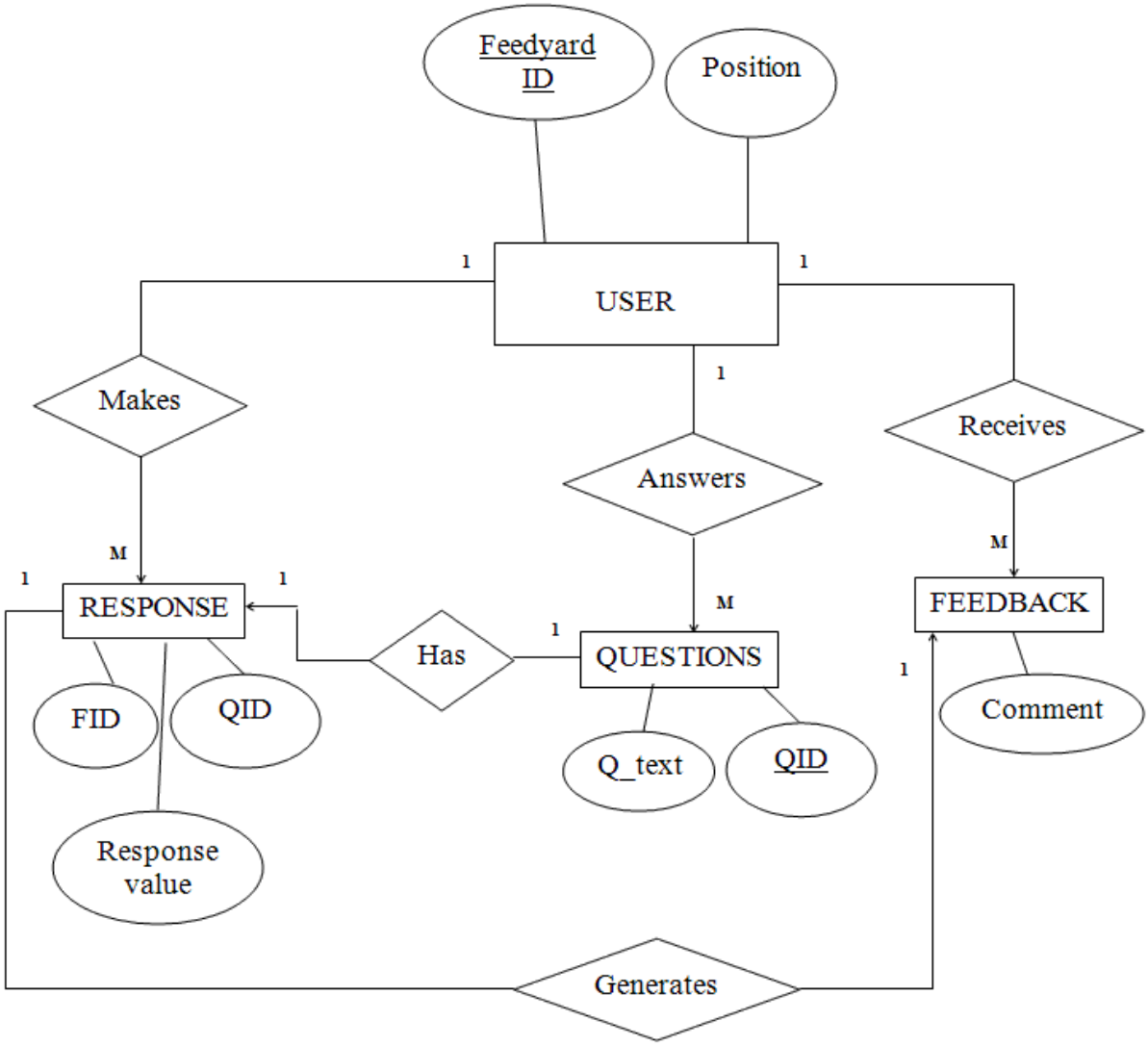

Figure 3.1 ER Diagram of the BRIC

3.2 Data Flow Diagrams

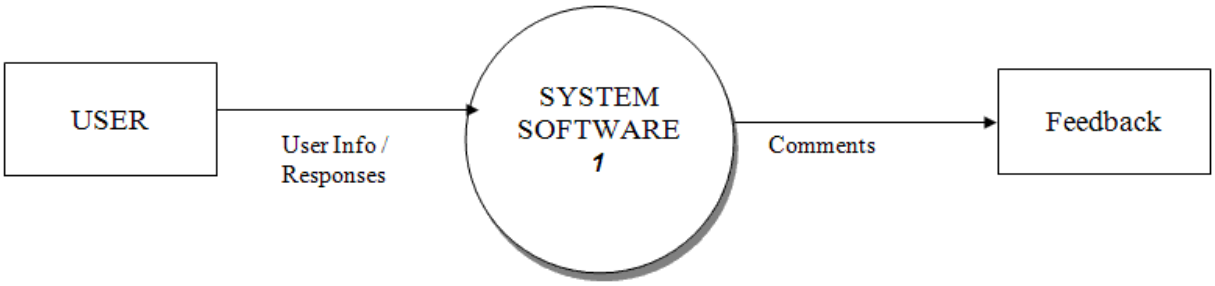

Figure 3.2 Context Level Data Flow Diagram

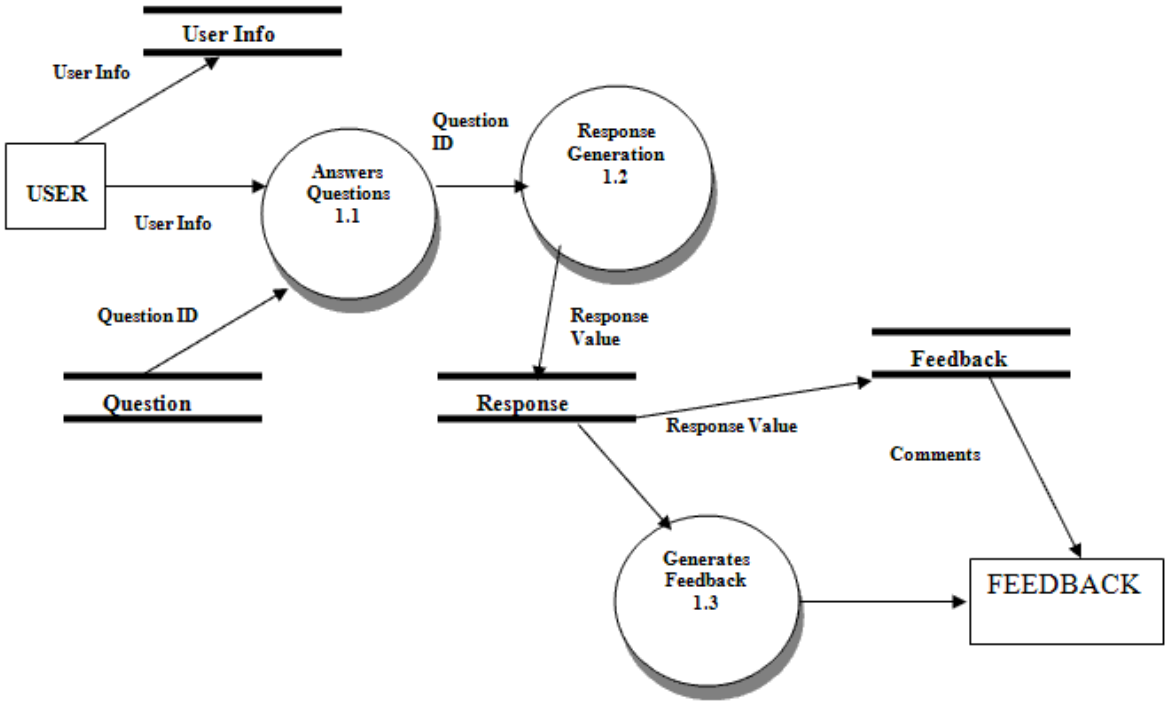

Figure 3.3 Level 1 Data Flow Diagram

3.3 State Transition Diagram

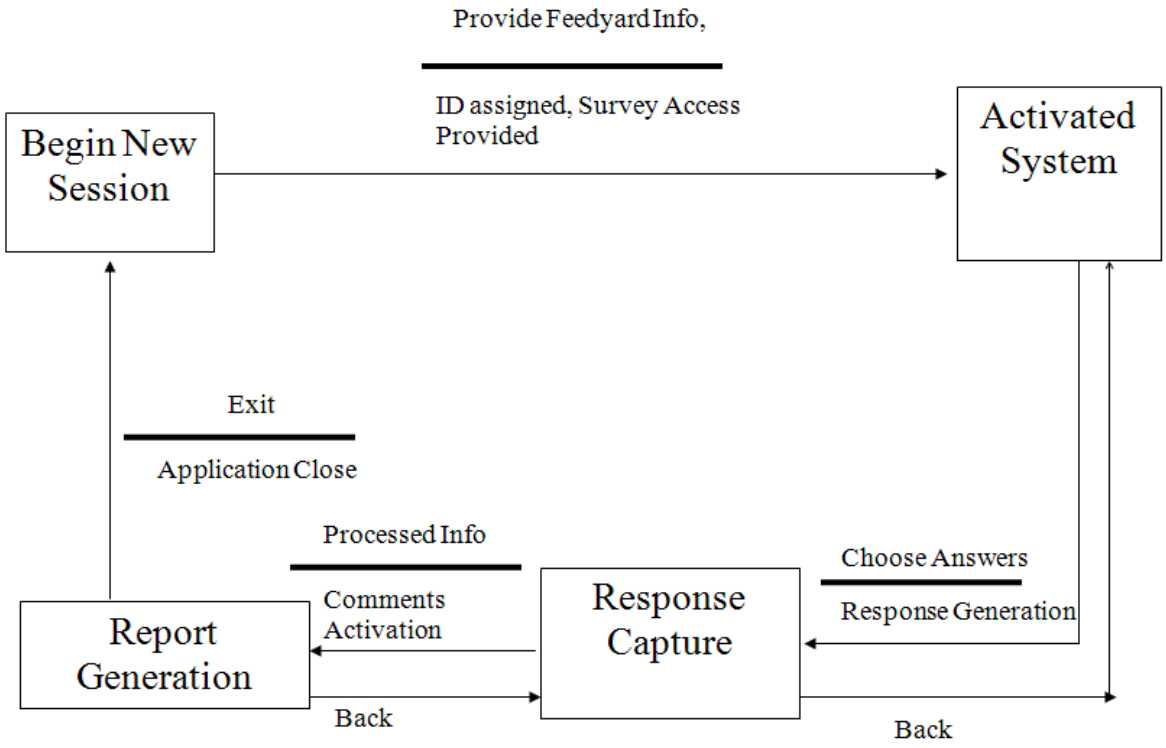

Figure 3.4 State Transition Diagram

During this phase of the development, I had a lot of discussions with the professor involved in the research. After collecting the information, I was able to create the ER diagram which is one of the most useful model forming tools to organize this application. ER diagram is a model that describes stored data of the system at a high level of abstraction. The processes and the flow of information were analyzed in detail which resulted into a detailed data flow diagram. Also, the states were analyzed and the transition of one state to another is depicted in the state transition diagram.

CHAPTER 4 - Design

4.1 Design Goals

The design of the web survey application involves the design of the following:

1. Design of a database schema which holds the user information, responses and can generate reports dynamically.
2. Design of a structure which helps the user to navigate from one part of the survey to the other at any point of time.
3. Design of interactive reports which should facilitate printing and saving at end user's machine.

4.2Architectural Design

4.2.1Architectural Context Diagram

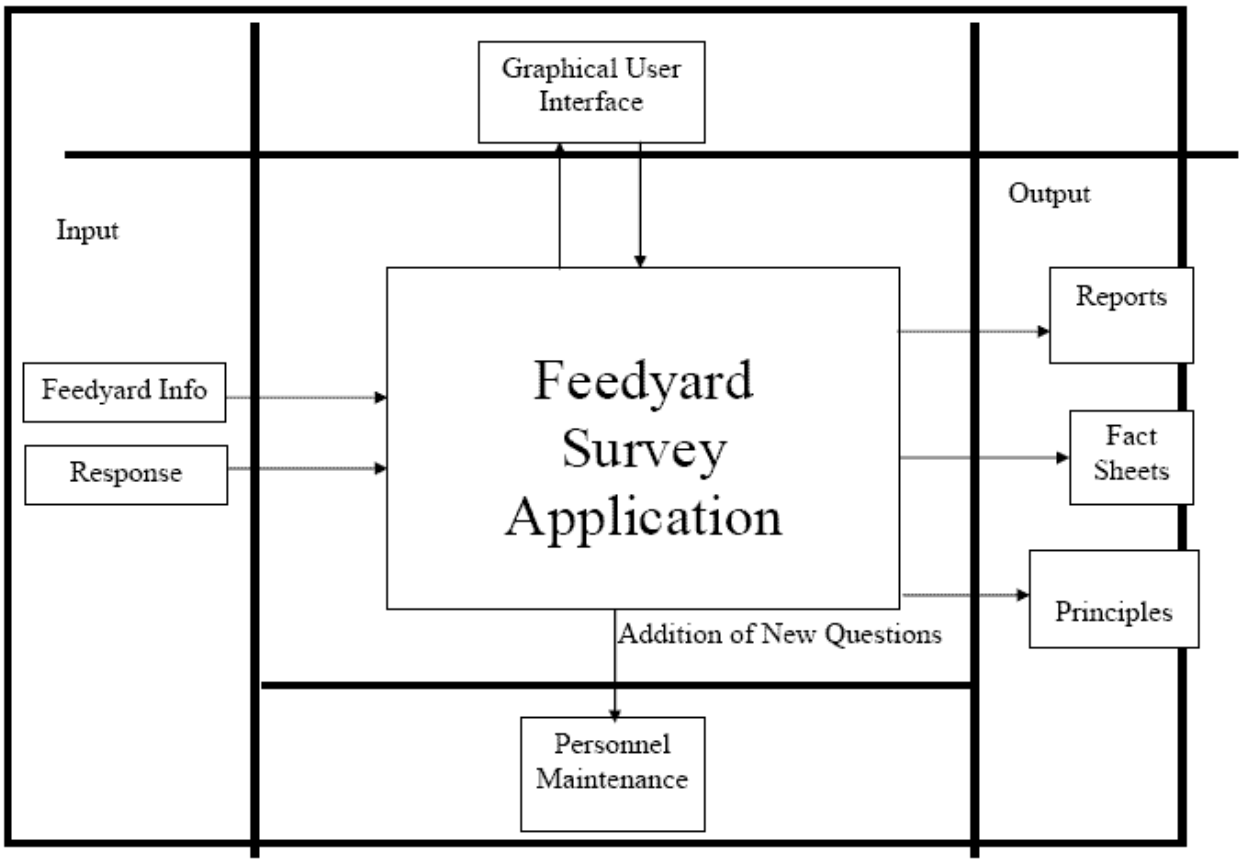

Figure 4.1 Architectural Context Diagram

4.2.2 Description of Architectural Design

In this context diagram, the information provided to and received from the 'Feedyard Survey Application' is identified. The arrows represent the information received or generated by the Feedyard Survey Application. The closed boxes represent the set of sources and sinks of information.

In the system, we can observe that the user interacts with the application through a graphical user interface. The inputs to the system are the feedyard information provided by the user and all the responses to the survey questions. Also, the output is in the form of reports which are based on the responses the user makes. There are some static outputs which do not change over the execution of the project such as principles and disease fact sheets.

Other than this, the maintenance of the system can be done by the administrator which can be addition, edition or deletion of more questions in the survey.

4.2.3 Component Diagram

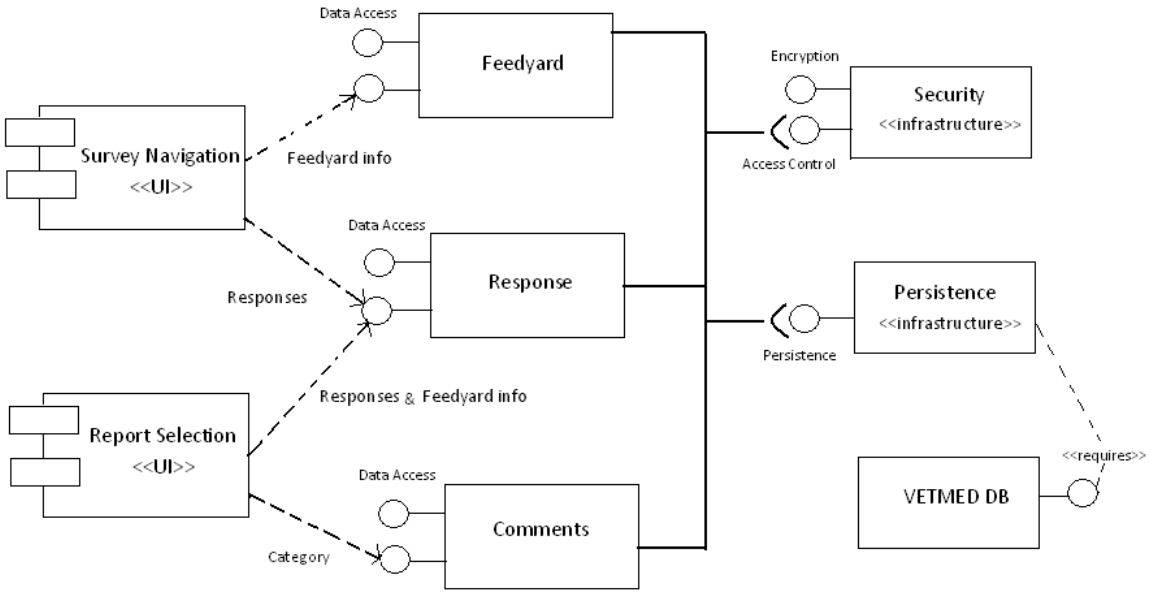

Figure 4.2 Component Diagram

4.3 Procedural /Modular Approach

Following are all the modules designed for the web survey application.

4.3.1 Initial Information Submission Module

This module starts with the generation of a 'Feedyard ID' for the user and the session of the user is maintained until the end of the application. The ID is an automatically generated primary key in the database and if any updates are made to the initial information they are updated using the same ID in the database.

4.3.2 User Navigation Module

This module allows the end user to navigate back and forth in the survey. Also if any section of the survey has already been completed the user can navigate to that section at any point of time. The module makes sure only those options are available to the end user for which the user is eligible.

4.3.3 Data Submission Module

This is the key module where user submits all his information. All the selections made during the survey are submitted in the database. After the data has been submitted, the user cannot make any changes. The user is prompted to wait until the reports are generated.

4.3.4 Report Generation Module

This is the module where the feedback for the user is generated in the form of reports. The user can select from various options to view the different category of reports. The user is given the option to convert the report to PDF or EXCEL and save it.

CHAPTER 5 - Implementation

5.1 Database Implementation

The design of the database was similar to the analysis phase. The database has been developed using MS Access 2007.

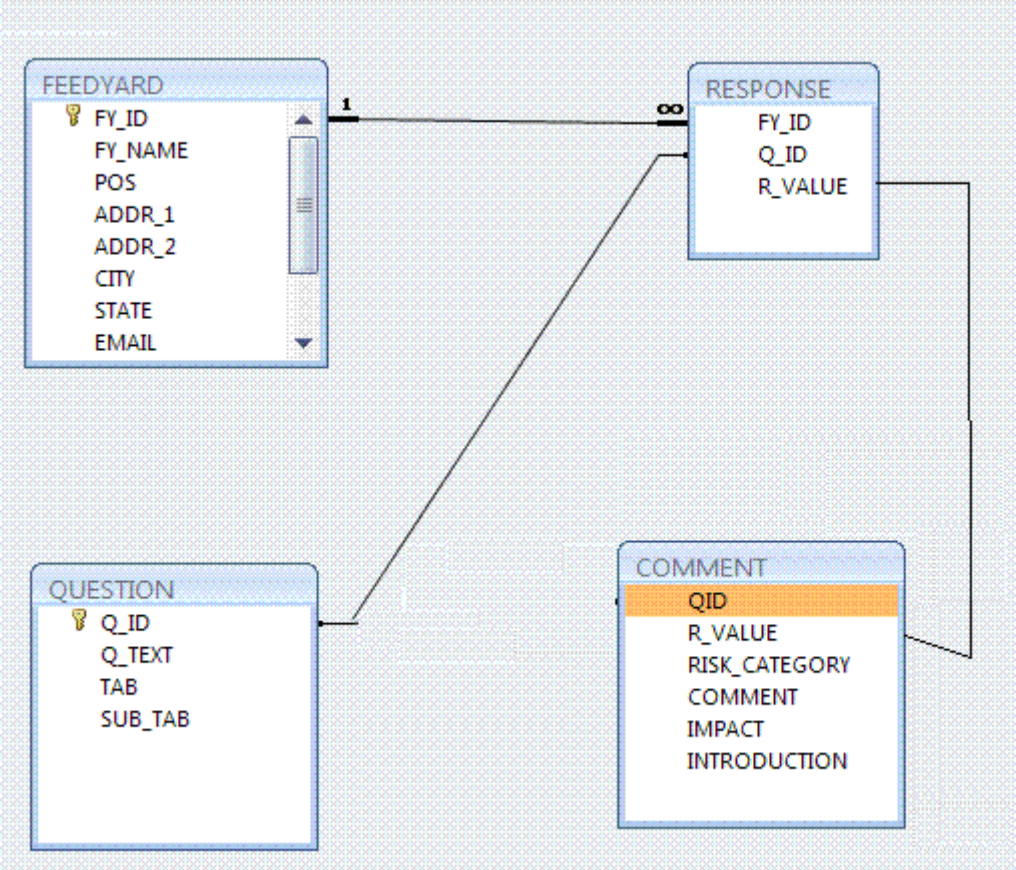

Figure 5.1 Database Implementation using MS Access

5.2 User Interface Design and Implementation

The user interface of the application has been designed using Microsoft Visual Studio 2005. The main controls used in the design are Multi View control, Menu Control, Report Viewer and these controls are provided with ASP.NET 2005. Following are the screenshots of the user interface.

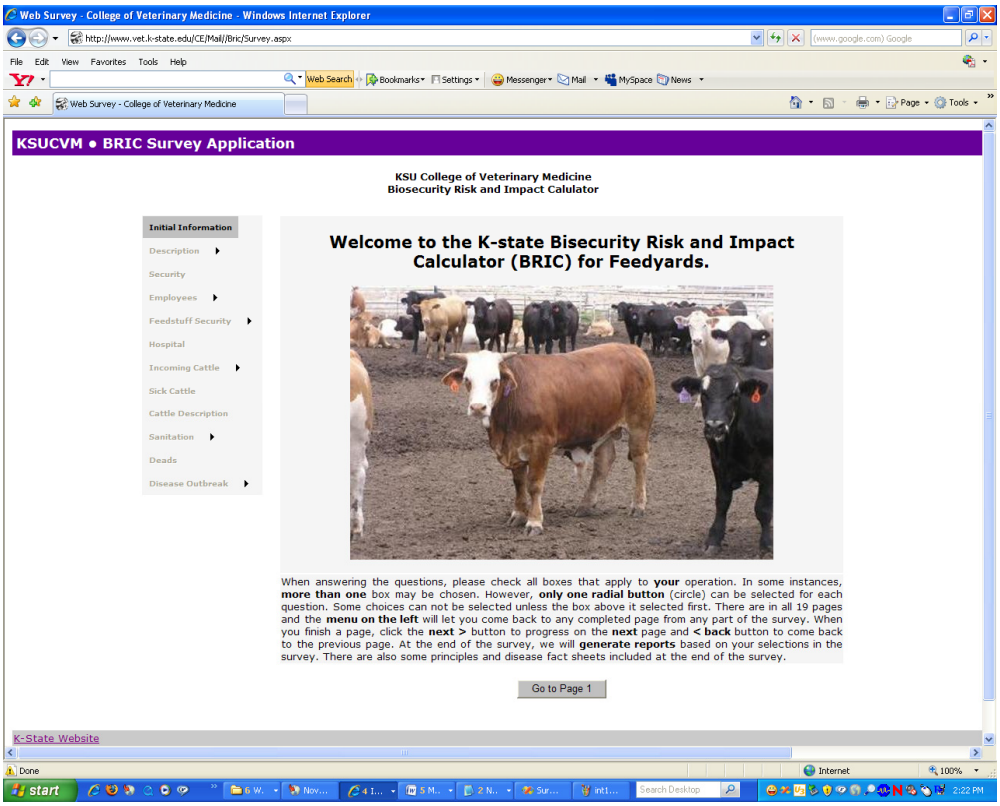

Figure 5.2 Instructions provided at the beginning and on the click of help button.

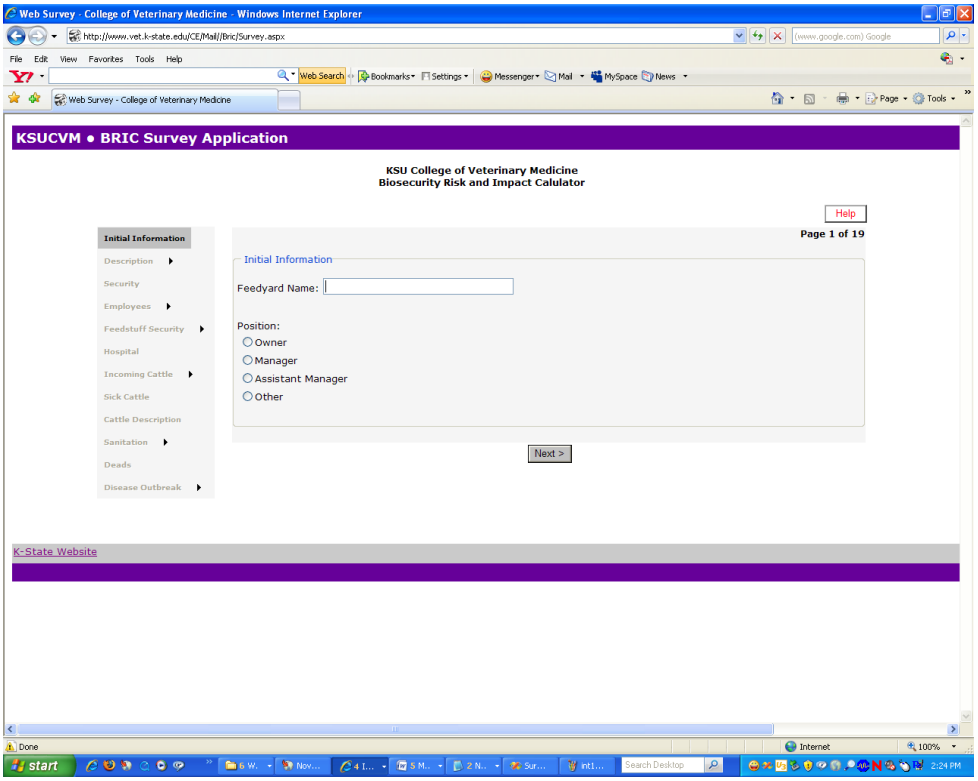

Figure 5.3 Initial Information Screen, user ID is generated here.

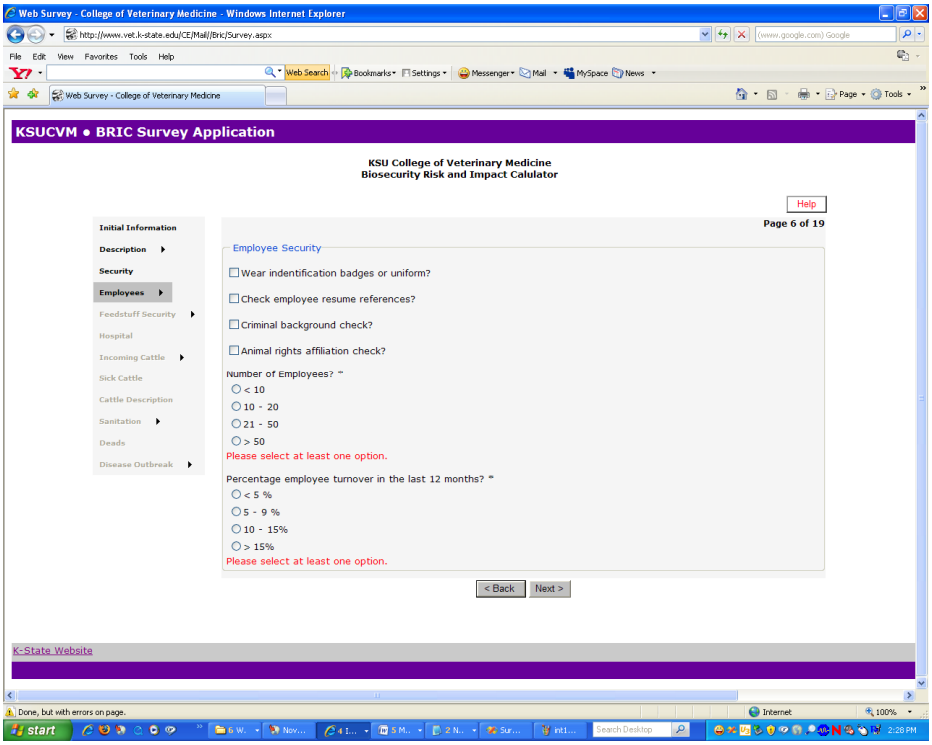

Figure 5.4 Validations used for compulsory questions.

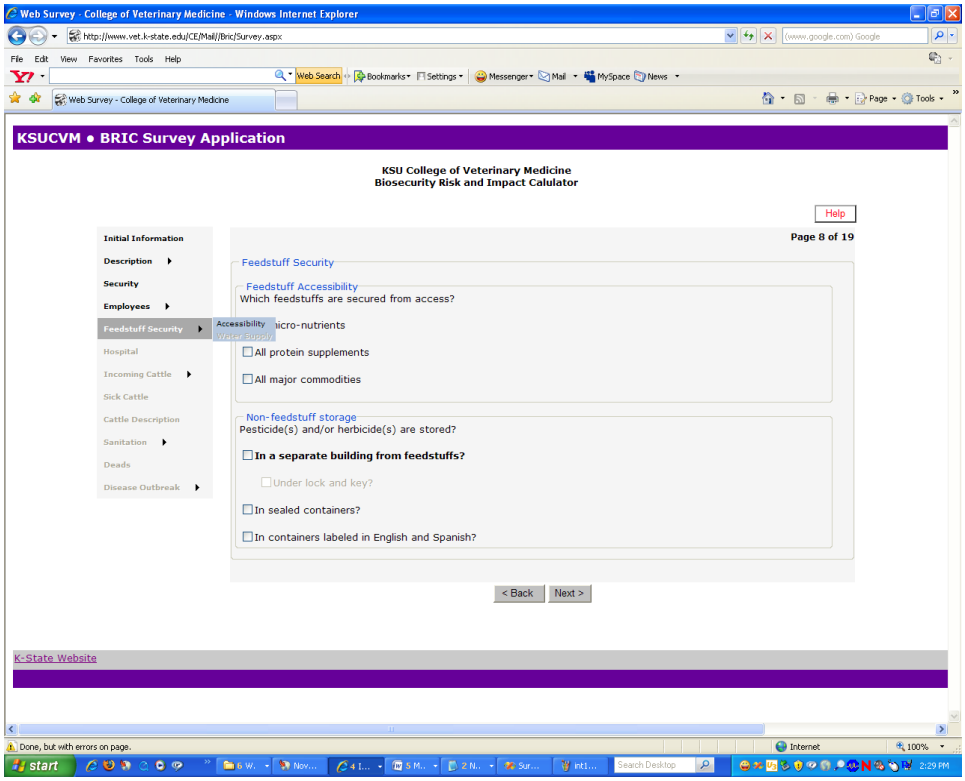

Figure 5.5 Menu has sub sections enabled only when the section has been completed.

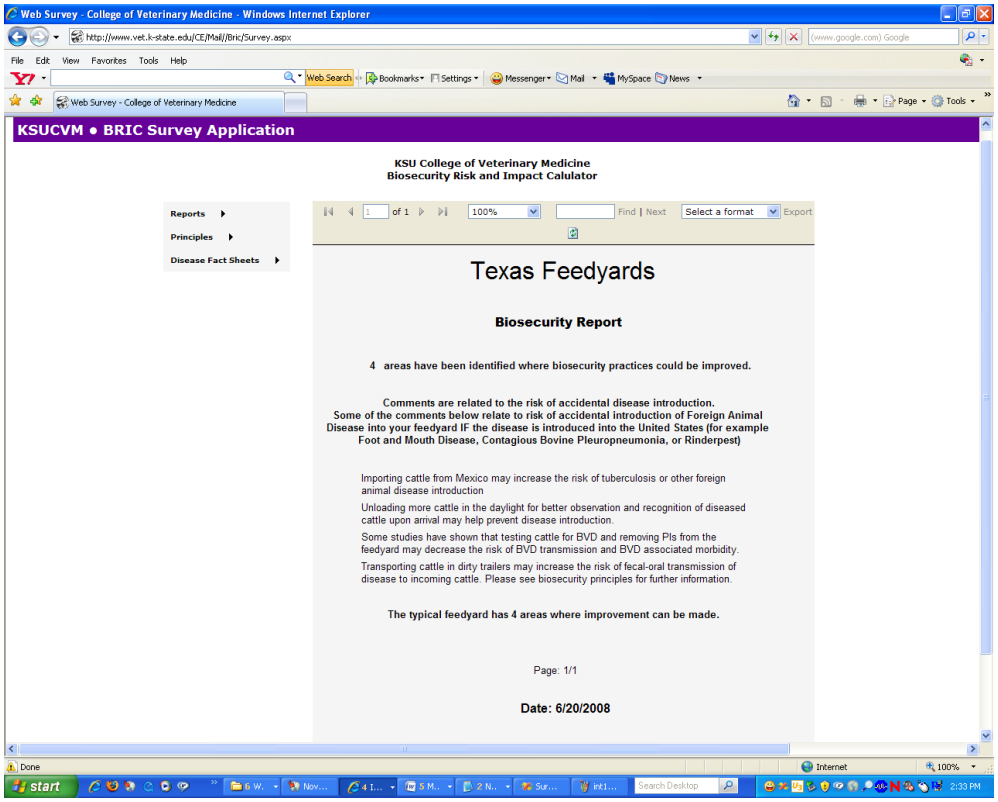

Figure 5.6 Reports generated with a different Menu to view different reports and options.

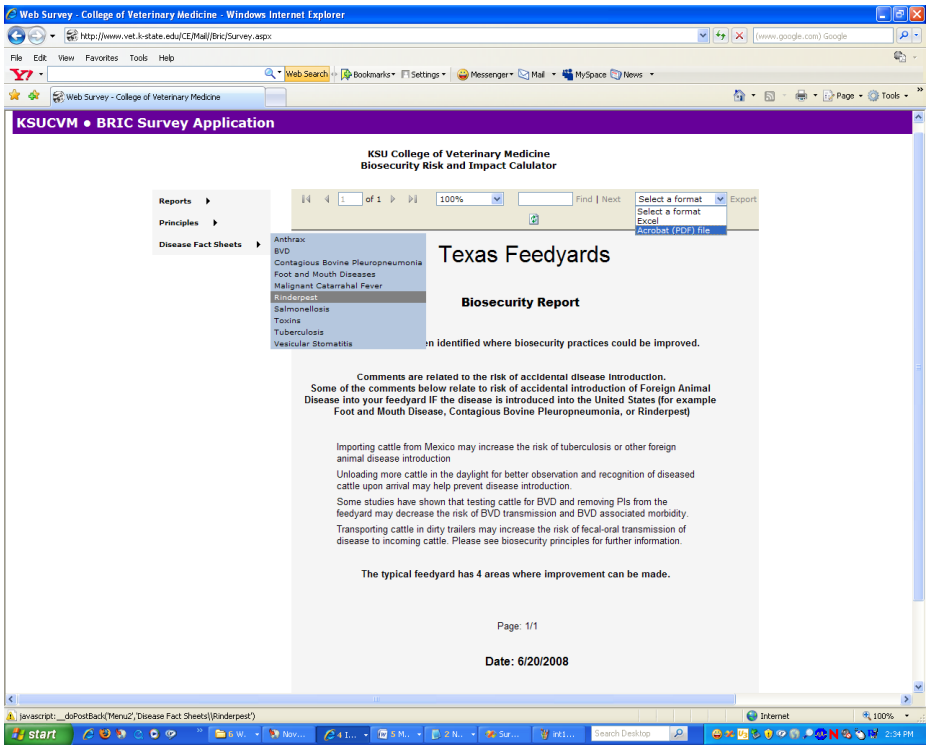

Figure 5.7 Reports can be saved using Microsoft Excel or Adobe PDF options.

5.3 Technical Discussions

The implementation of the database application has a table named 'QUESTIONS'. This database table consists of all the questions in the survey. But the user interface implementation still consists of questions which are static and are not retrieved from the database. The reason for this redundant implementation is to improve the navigation between the different categories of questions. If data is retrieved from the database, the speed of the navigation is affected and the navigation from one part of the survey to the other becomes comparatively slower. As the survey consists of 19 views, the slower navigation affects the performance of the application and increases the survey completion time for the user.

Also, as per the specifications of the application from the client, the questions of the survey will not be changed in the near future. So keeping the questions static on the user interface was a satisfactory implementation to improve the performance of the navigation of the survey. If in the near future, there are a huge number of questions added in the survey, a better approach would be to retrieve the questions from the database dynamically.

The application consists of 4079 lines of code and the time taken to complete the application is around 5 months. 15 hours per week were spent for the development.

CHAPTER 6 - Testing

Software testing is a process of running with intent of finding errors in software. Software testing assures the quality of software and represents final review of other phases of software like specification, design, code generation etc.

6.1 Unit Testing

Unit testing emphasizes the verification effort on the smallest unit of software design i.e.; a software component or module. Unit testing is a dynamic method for verification, where program is actually compiled and executed. Unit testing is performed in parallel with the coding phase. Unit testing tests units or modules not the whole software.

I have tested each view of the application individually. As the modules were built up testing was carried out simultaneously, tracking out each and every kind of input and checking the corresponding output until module is working correctly.

The functionality of the modules was also tested as separate units. As we have mentioned, the user navigation module works along with the other modules, it was also tested independently without considering the other modules. The next button and the back button functionality was an important way to test as they fall under the separate cases in the logic. Each individual case was tested independently which made sure that each view of the survey was working as a separate unit.

Also, as the user data submission module submits all the information gathered from the survey into the database, responses from each view were tested independently. As the survey was being implemented, it was made sure that responses to all the questions are recorded into the database. An individual submission with each question has been tested for the response capture.

The report generation module has been tested as an independent unit. A user is provided with many selections options in order to view the reports. While configuration of the reports, each individual report was independently tested for its functionality. It was made sure that the desired data related to a particular report is retrieved from the database.

6.2 Integration Testing

In integration testing a system consisting of different modules is tested for problems arising from component interaction. Integration testing should be developed from the system specification. Firstly, a minimum configuration must be integrated and tested.

In my project I have done integration testing in a bottom up fashion i.e. in this project I have started construction and testing with atomic modules. After unit testing the modules are integrated one by one and then tested the system for problems arising from component interaction.

6.3 Validation Testing

It provides final assurances that software needs all functional, behavioural & performance requirement. Black box testing techniques are used.
There are three main components

- Validation test criteria (no. in place of no. & char in place of char)
- Configuration review (to ensure the completeness of s/w configuration.)
- Alpha & Beta testing-Alpha testing is done at developer's site i.e. at home & Beta testing once it is deployed.

Test Cases- I have used a number of test cases for testing the product. There were different cases for which different inputs were used to check whether desired output is produced or not.

1. Proper run time generation of the Feedyard ID.
2. Correct reports are generated at the end of the survey for this ID.
3. With the back button, if changes are made, they are reflected correctly in the application.
4. Each question's response is reflected to the correct question ID in the database.
5. Options within the Menu are enabled once the specific page has been completed.
6. Back and Next button navigates the user to the correct and desired page.
7. Once submitted user should not be able to make changes.

6.4 White Box Testing

In white box testing knowing the internal working of the product, tests can be conducted to ensure that internal operations are performed according to specification and all internal components have been adequately exercised. In white box testing logical path through the software are tested by providing test cases that exercise specific sets of conditions and loops. Using white-box testing software developer can derive test case that

- Guarantee that all independent paths within a module have been exercised at least once.
- Exercise all logical decisions on their true and false side.
- Exercise all loops at their boundaries and within their operational bound.
- Exercise internal data structure to ensure their validity.

At every stage of project development I have tested the logics of the program by supplying the invalid inputs and generating the respective error messages. All the loops and conditional statements are tested to the boundary conditions and validated properly.

6.5 Stress and Performance Testing

Performance testing is an essential element in successfully deploying a Web application. It's important to understand how the application and the Web server would behave as more and more users visit the Web site. In order to simulate that type of usage for a Web application, we would either need to coordinate with hundreds or even thousands of real users to access our Web site within a designated period of time or work with a testing tool that can reproduce such user loads.

Many Web performance testing tools are available to help. Basically, these tools allow us to use a minimal number of client computers to simulate a large number of virtual users, concurrently requesting predefined pages of the Web site. Each of these virtual users emulates the exact communication protocols between a real Web browser and the Web server.

The tool that has been used in this application is called the Web Application Stress Tool - WAS from Microsoft. The test script was created which completes the survey and the settings

were changed to simulate 100 users hitting the site for 1 hour and performance was tested with a low dial up bandwidth. The steps and the results have been demonstrated as follows:

The concept behind WAS is simple: We can create a test script by capturing a browser session using Internet Explorer basically walking through our application, as a typical user would do. As we do this WAS captures the content of all these Web requests. WAS captures everything: Hyperlink clicks, Form submissions, Redirect links and everything needed to capture the user's session through our site. We can use the Browser Recorder to capture a browser session and have WAS generate a test script from captured links. There are quite a few options we can choose for the Browser Recorder: Capture delay, and record cookies and host headers. The delay between requests will result a more realistic test in terms of how people are actually navigating a site, giving us a more accurate picture of how users on a site map to connections on the Web server. When we click Next|Finish on the browser recorder we are whisked into IE and ready to capture requests in our browser. Once we are done, we can switch back to the running WAS application in the background and click on the Stop Recording button (Figure 6.1).

If we look closely at the WAS form before clicking the 'Stop Recording' button we can see how WAS is capturing the browsers progress. The data is captured and stored in an Access (MDB) database file including any content captured from form variables.

Once we have captured the script we can see that our captured Web links are shown on the right in the data view of the main WAS window. With the links captured our next step is to configure the load options for running the script. We do this using the Settings option in the list and we can see a dialog as shown in Figure 6.2.

After running the test script with 100 concurrent users and for 1 continuous hour, there were no performance issues and no socket errors and all essential details are obvious from the above observations generated from the WAS tool.

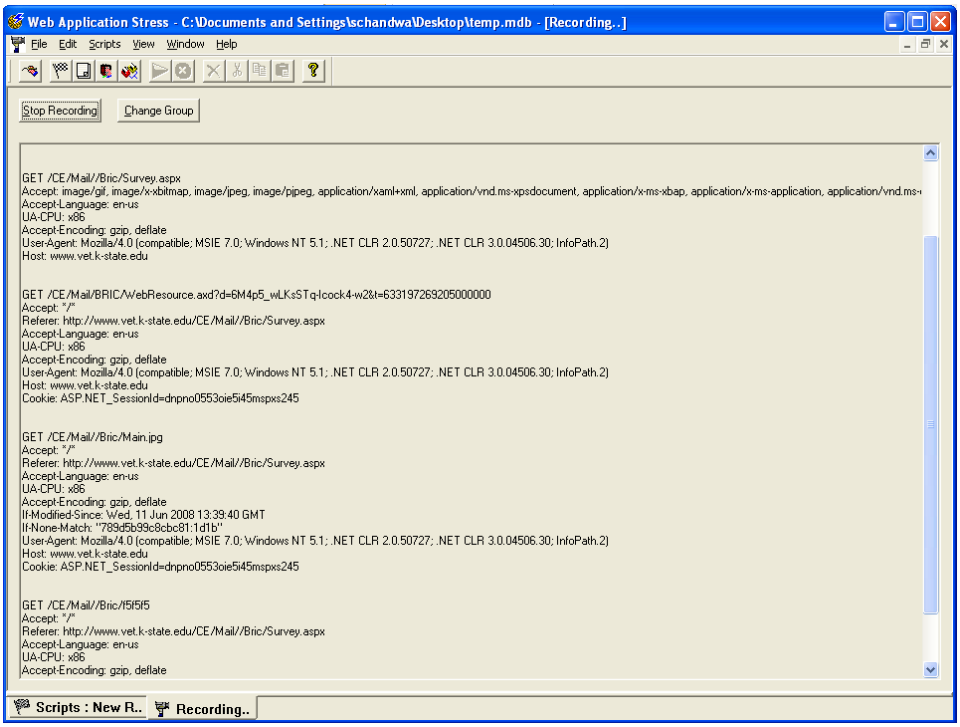

Figure 6.1 While recording a script WAS monitors IE to capture all incoming data and stores it into a database file.

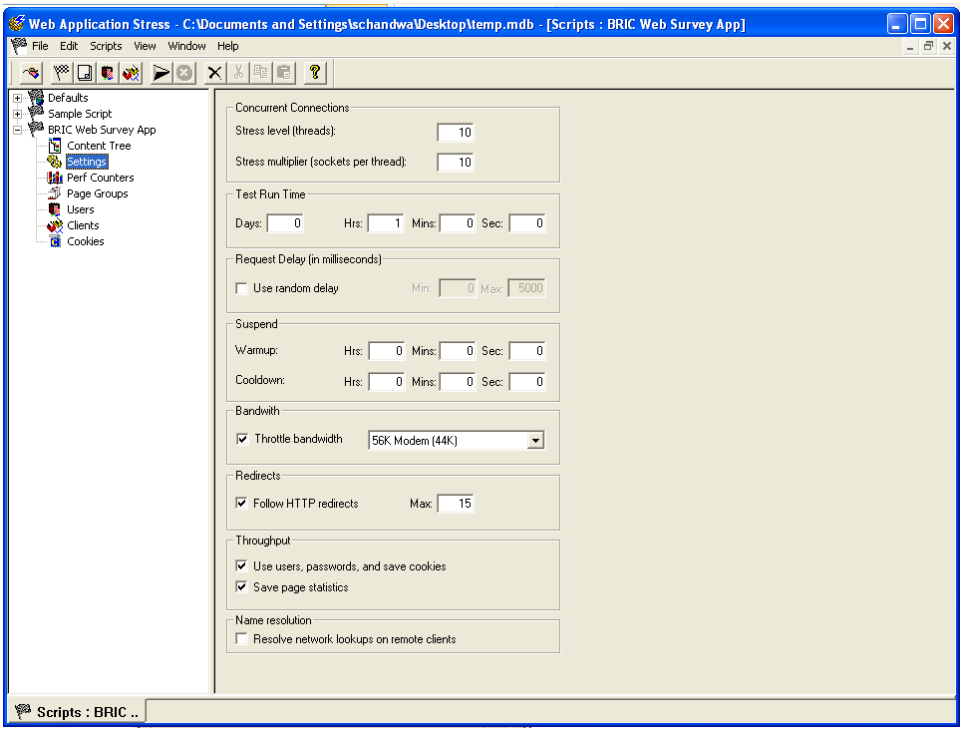

Figure 6.2 We can specify the number of simulated clients by setting the number of threads and number of sockets on each thread

Stress Level

This property determines the number of threads that will be run by WAS to hit the client application.

Stress Multiplier

This property determines the number of sockets that are created on each of the above threads. The end result is that the Stress Level times the Stress Multiplier equal the number of clients you are simulating. Threads * Sockets = Total clients.

Test Run Time

This option allows you to specify how long to run the test. This is great to start up a test and let it run for exactly 1 hour for example, to see exactly how continuous pounding will affect performance.

Request Delay

The request delay allows you to provide more realistic user simulation, since users don't continuously click on links as soon as a page loads. Typically users look around a page, find a link and then click it. Even a familiar user may take 5 seconds between requests – new users will take much longer.

Throttle Bandwidth

This option causes WAS to monitor the traffic being generated both on the outgoing on incoming links and optionally allows limiting the bandwidth available.

My goal here however is to see how well the backend performs and I try to actually run as many hits as I possibly can before the system becomes to loaded: CPU close to 100% and pages returned taking more than 10 seconds from another machine.

With 100 continuous clients I'm not even close to the 100% mark: 35% CPU utilization and when hitting the server with a separate browser any requests are returned immediately. So, I double the count to 200 clients (20 threads/10 sockets). Now things get more interesting – the CPU is running at 75% average with occasional spikes close to 100%.

In order to truly test operation under load we need to stress test for long periods. I like to run my tests for at least 1 hour. Applications tend to get more resource hungry the longer they run – it's not uncommon to see slowdowns over long periods of hard operation.

In all fairness, though, WAS does provide the ability to log NT Performance counters from the Web server to allow logging of server performance statistics over time. WAS generates a file hcounters.csv which contains these counter values, which you can then manipulate and graph externally (in tools such as Excel for example).

Table 6-1 The following report was generated by the WAS tool for testing the performance.

Server:	129.130.129.21
Number of threads:	100
Test length:	01:00:00
Warmup:	00:00:00
Cooldown:	00:00:00
Follow Redirects:	Yes
Max Redirect Depth:	15
Overview	
Checking the performance	
Number of test clients:	1
Number of hits:	30318
Requests per Second:	8.28
Socket Statistics	
Socket Connects:	30318
Total Bytes Sent (in KB):	37460.07
Bytes Sent Rate (in KB/s):	10.41
Total Bytes Recv (in KB):	385202.29
Bytes Recv Rate (in KB/s):	107.00
Socket Errors	
Connect:	0
Send:	0
Recv:	0
Timeouts:	0

CHAPTER 7 - Results

The application is being used by researchers at K-state to perform analysis over the data collected using the survey. As there are three main reports generated by the application, a brief introduction about the biosecurity, biocontainment and security practices have been given here along with the analysis and calculations made over all the three practices.

7.1 Biosecurity Practices

Biosecurity was defined earlier as management strategies for prevention of disease entry which are different for each animal production system. Feedyards accept various degrees of risk depending on the different sources of cattle imported, but observing cattle at arrival will help determine their arrival state of health and may reduce the risk of importing disease into the feedyard.

Traditional biosecurity involves controlling introduction of disease by quarantine and testing of imports prior to introduction to the resident herd. During a quarantine period, animals should be monitored for signs of illness, tested and vaccinated to match the immune status of the herd. If animals cannot be kept on another site, they should be kept on the edge of the premises away from contact with other cattle. Most feedyards keep cattle in the receiving facility until they are processed and within three days they are introduced to the rest of the cattle and placed in their home pen. The idea of quarantine may not be well accepted because facility design will not support separation of cattle for a long period of time when the optimal goal is to keep every pen full.

A quarantine period also allows time for testing cattle for potential high risk diseases. Testing cattle on arrival at the feedyard is impractical for most diseases with the possible exception of bovine viral diarrhea virus (BVDV). This disease will be discussed in further detail later. However, feedyards will likely continue to import cattle that have been commingled with other cattle at the auction market without testing or a period of isolation.

Table 7-1 shows the responses made related to biosecurity practices in 106 surveys and the analysis and calculations used by researchers at K-state.

Table 7-1 Proportion of 106 feedyards in 5 Central Plains states at which practices relevant to biosecurity were implemented

	Affirmative responses		No. of cattle for 1-time capacity							
			1,000 to 3,999 (n = 29)		4,000 to 15,999 (n = 32)		16,000 to 31,999 (n = 25)		≥ 32,000 (n = 20)	
Question (No. of responses)	No.	%	No.	%	No.	%	No.	%	No.	%
Are > 50% of cattle arriving from the auction market? (106)	61	57.6	18	62.1	18	56.3	13	52.0	12	60.0
Are trailers required to be cleaned before loading incoming cattle?* (106)	15	14.2	8	27.6	5	15.6	0	0	2	10.0
Are visitors required to use feedyard vehicles when on the facility? (106)	42	39.6	13	44.8	12	37.5	10	40.0	7	35.0
Is history of animal contact collected from visitors or vendors? (106)	3	2.8	0	0	1	3.1	0	0	2	10.0
Is history of international travel collected from visitors or vendors? (106)	5	4.7	0	0	1	3.1	0	0	4	20.0
Are clean boots or foot coverings required for visitors? (106)	2	1.9	0	0	1	3.1	0	0	1	5.0
Are all cattle with unknown BVDV status ear notched and tested so persistently infected cattle can be removed from the feedyard? (106)	3	2.8	0	0	1	3.1	1	4.0	1	5.0
Are new arrivals separated from other cattle until BVDV persistently infected status is known? (3)	2	66.7	ND	ND	0	0	1	100	1	100
Are carcasses disposed of by a rendering service? (106)	92	86.8	23	79.3	27	84.4	23	92.0	19	95.0
Do rendering trucks drive across the regular feedyard traffic pattern to access dead cattle? (92)	75	81.5	17	73.9	24	88.9	19	82.6	15	78.9
Does the feedyard have a written outbreak response plan for FMD?* (106)	34	32.1	5	17.2	7	21.9	13	52.0	9	45.0

Questions with < 106 responses are the result of dependence on the response to other questions in the survey tool.
ND = Not determined.
See Table 2 for remainder of key.

7.2 Biocontainment Practices

Biocontainment is achieved by implementing strategies to reduce risk associated with the transmission of pathogenic agents among cattle within a feedyard. The large number of animals and relatively high population density in modern feedyards make biocontainment an important issue. Disease of cattle within feedyards is inevitable, but it can be managed with strategies such as segregation of sick animals from healthy animals or cleaning and disinfection of equipment and facilities to decrease exposure of susceptible cattle. These principles apply to many diseases endemic to cattle in US feedyards.

Table 7-2 shows the responses made related to biocontainment practices in 106 surveys and the analysis and calculations used by researchers at K-state.

Table 7-2 Proportion of 106 feedyards in 5 Central Plains states at which practices relevant to biocontainment were implemented.

			No. of cattle for 1-time capacity							
	Affirmative responses		1,000 to 3,999 (n = 29)		4,000 to 15,999 (n = 32)		16,000 to 31,999 (n = 25)		≥ 32,000 (n = 20)	
Question (No. of responses)	No.	%	No.	%	No.	%	No.	%	No.	%
Are cattle treated and separated in hospital pens? (106)	95	89.6	24	82.8	28	87.5	25	100	18	90.0
Do sick cattle in hospital pens have fence-line contact with healthy cattle? (95)	35	36.8	7	29.2	11	39.3	9	36.0	8	44.4
Are feedyard loaders and trucks used to handle dead cattle and manure ever used for feed handling?* (106)	27	25.5	13	44.8	9	28.1	3	12.0	2	10.0
Is equipment always thoroughly cleaned and disinfected after handling dead cattle or manure before handling feed? (27)	4	14.8	2	15.4	1	11.1	0	0	1	50.0
Is the equipment used for oral administration of treatments ever cleaned?* (106)	98	92.5	23	79.3	31	96.9	25	100	19	95.0
Is the equipment used for oral administration of treatments cleaned after each animal? (106)	48	45.3	10	34.5	15	46.9	13	52.0	10	50.0
Is the equipment used for oral administration of treatments ever disinfected?* (106)	68	64.2	13	44.8	18	56.3	22	88.0	15	75.0
Is the equipment used for oral administration of treatments cleaned and disinfected after each animal?* (106)	28	26.4	2	6.9	7	21.9	11	44.0	9	45.0
Are cattle unloaded in the same facility used to treat sick cattle?* (106)	38	35.9	22	75.9	13	40.6	1	4.0	2	10.0
Is the loading-unloading facility cleaned weekly or more often? (106)	31	29.2	7	24.1	11	10.4	5	20.0	8	40.0
Is the loading-unloading facility cleaned and disinfected weekly or more often? (106)	2	1.9	1	3.4	1	3.4	0	0	1	3.4
Is the facility used for both treatment and unloading disinfected weekly or more often? (38)	1	2.6	1	4.5	0	0	0	0	0	0
Is the processing facility cleaned daily?* (106)	51	48.1	3	10.3	12	37.5	18	72.0	18	90.0
Is the processing facility cleaned and disinfected daily? (106)	11	10.4	0	0	3	9.4	5	20.0	3	15.0
Is the treatment facility cleaned daily?* (106)	43	40.6	2	6.9	10	31.3	16	64.0	15	75.0
Is the treatment facility cleaned and disinfected daily? (106)	12	11.3	0	0	3	9.4	6	24.0	3	15.0

Questions with < 106 responses are the result of dependence on the response to other questions in the survey tool.
*Values differ significantly ($P < 0.05$) among feedyard strata.
n = Number of feedyards in the NASS stratum.

7.3 Security Practices

Security begins with identifying the factors posing the biggest threat in terms of both likelihood and impact. Threats to the feedyard may include everything from theft of feed, supplies or cattle by disgruntled neighbors, employees or activist groups intending to make a statement of their ideology. Potential threats should be evaluated in reference to the likelihood of occurrence.

Table 7-3 shows the responses made related to security practices in 106 surveys and the analysis and calculations used by researchers at K-state.

Table 7-3 Proportion of 106 feedyards in 5 Central Plains states at which practices relevant to feedyard security were implemented.

Question (No. of responses)	Affirmative responses		No. of cattle for 1-time capacity							
			1,000 to 3,999 (n = 29)		4,000 to 15,999 (n = 32)		16,000 to 31,999 (n = 25)		≥ 32,000 (n = 20)	
	No.	%	No.	%	No.	%	No.	%	No.	%
Does the feedyard have a perimeter fence?* (106)	67	63.2	8	27.6	20	62.5	20	80.0	19	95.0
Will the perimeter fence stop vehicles?* (67)	45	67.2	2	25.0	14	70.0	14	70.0	15	78.9
Will the perimeter fence stop humans?* (67)	10	14.9	0	0	2	10.0	3	15.0	5	26.3
Are all cattle behind locked gates?* (106)	43	40.6	5	17.2	9	28.1	13	52.0	16	80.0
Are all micronutrients secured from unauthorized access?* (106)	47	44.3	7	24.1	10	31.3	18	72.0	12	60.0
Are all protein supplements secured from unauthorized access? (106)	44	41.5	10	34.5	10	31.3	15	60.0	9	45.0
Is a night watchman employed?* (106)	35	33.0	0	0	7	21.9	12	48.0	16	80.0
Are there signs directing visitors to check in at the office?* (106)	44	41.5	0	0	9	28.1	17	68.0	18	90.0
Is a visitor log maintained and enforced?* (106)	25	23.6	0	0	3	9.4	7	28.0	15	75.0
Are employee résumé references checked at hiring?* (106)	80	75.5	18	44.8	28	87.5	24	96.0	15	75.0
Is a criminal background check performed at hiring?* (106)	28	26.4	3	10.3	8	25.0	12	48.0	5	25.0

Questions with < 106 responses are the result of dependence on the response to other questions in the survey tool.
See Table 2 for remainder of key.

CHAPTER 8 - Conclusion

The 'BRIC - Biosecurity Risk and Impact Calculator' is designed to provide a platform through which researchers and feedyard managers can make an improvement over the existing condition of the feedyards. I hope that the information collected through this survey application will be beneficial for a lot of data analysis by the researchers. Also, there can be a significant improvement if the feedyard managers work upon the measures generated in the form of reports through this application.

The survey application development has been a learning process as it has been implemented using the .NET framework. Reporting and navigation has been implemented in an efficient manner which makes the application easy to use for the end user. The application has been created for the College of Veterinary Medicine which is a totally different environment for a software developer. Thus the application taught me to understand the specifications of veterinary science and work accordingly. Also the tools used for performance testing were from Microsoft which was again a learning process.

8.1 Limitations

The survey does not incorporate a way to add, edit or delete the questions through a user interface. For this reason, the maintenance part of the survey needs a programmer who should be familiar with latest technology using which the survey has been designed.

8.2 Scope for Future Work

The application can be modified to incorporate an administrative module. This module can provide the administrator with a user interface through which modifications can be made in the survey. This can add the functionality of editing the survey using the user interface which will eliminate the need of a programmer to maintain the survey.

References

[1] Aric Brandt, Feedyard biocontainment, biosecurity, and security risks and practices of central plains feedyards
http://krex.k-state.edu/dspace/bitstream/2097/343/1/AricBrandt2007.pdf

[2] Microsoft Report Viewer Control
http://msdn.microsoft.com/enus/library/microsoft.reporting.webforms.reportviewer(VS.80).aspx

[3] Microsoft Menu Control
http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.menu.aspx

[4] Microsoft Multi View Control
http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.multiview.aspx

[5] Software Testing
http://en.wikipedia.org/wiki/Software_testing